Security Requirements Engineering

Nancy R. Mead, Software Engineering Institute [vita[3]]

2006-02-21

Security requirements are often identified during the system life cycle. However, the requirements tend to be general mechanisms such as password protection, firewalls, virus detection tools, and the like. As a result, security requirements that are specific to the system and that provide for protection of essential services and assets are often neglected. Several approaches to security requirements engineering are described that can help organizations ensure that their products effectively meet security requirements.

Overview (Importance of Requirements Engineering)

Related Links

- Comprehensive, Lightweight Application Security Process (CLASP[7])
- System Quality Requirements Engineering (SQUARE[8])
- Core security requirements artifacts[9]
- Attack trees[10]
- Misuse/abuse cases[11] and process diagram[12]
- REVEAL[13]
- Software Cost Reduction (SCR[14])
- Common Criteria[15]
- Bibliography[16] for requirements engineering

It is well recognized in industry that requirements engineering is critical to the success of any major development project. Several authoritative studies have shown that requirements engineering defects cost 10 to 200 times as much to correct once fielded than if they were detected during requirements development [Boehm 88[17], McConnell 01[18]]. Other studies have shown that reworking requirements, design, and code defects on most software development projects costs 40 to 50 percent of total project effort [Jones 86[19]], and the percentage of defects originating during requirements engineering is estimated at more than 50 percent. The total percentage of project budget due to requirements defects is 25 to 40 percent [Wiegers 03[20]].

A recent study found that the return on investment when security analysis and secure engineering practices are introduced early in the development cycle ranges from 12 to 21 percent, with the highest

3. daisy:230 (Mead, Nancy)
7. daisy:229 (Comprehensive Lightweight Application Security Process (CLASP))
8. daisy:232 (SQUARE Process)
9. http://computing-reports.open.ac.uk/index.php/2004/200423
10. daisy:236 (Attack Trees)
11. daisy:125 (Misuse and Abuse Cases: Getting Past the Positive)
12. daisy:1475 (Create Misuse and Abuse Cases Process Diagram)
13. http://www.praxis-his.com/reveal/index.htm
14. http://www.software.hrl.com/sis/3-4/scr.html
15. daisy:239 (The Common Criteria)
16. daisy:231 (Requirements Engineering Annotated Bibliography)
17. #refs
18. #refs
19. #refs
20. #refs

rate of return occurring when the analysis is performed during application design [Soo Hoo 01[21]]. The National Institute of Standards and Technology (NIST) reports that software that is faulty in security and reliability costs the economy $59.5 billion annually in breakdowns and repairs [NIST 02[22]]. The costs of poor security requirements show that even a small improvement in this area would provide a high value. By the time that an application is fielded and in its operational environment, it is very difficult and expensive to significantly improve its security.

Requirements problems are among the top causes [Charette 05[23]] of why projects

- are significantly over budget
- are significantly past schedule
- have significantly reduced scope
- deliver poor-quality applications
- are not significantly used once delivered
- are cancelled

Requirements engineering typically suffers from the following major problems:

- Requirements identification typically does not include all relevant stakeholders and does not use the most modern or efficient techniques.
- Requirements analysis typically is either not performed at all (identified requirements are directly specified without any analysis or modeling) or analysis is restricted to functional requirements aimed at the end user, ignoring quality requirements, other functional and non-functional requirements, and architecture, design, implementation, and testing constraints.
- Requirements specification is typically haphazard, with specified requirements being ambiguous, incomplete (e.g., non-functional requirements are often missing), inconsistent, not cohesive, infeasible, obsolete, neither testable nor capable of being validated, and not usable by all of their intended audiences.
- Requirements management is typically weak with poor storage (e.g., in one or more documents rather than in a database or tool) and missing attributes, and is limited to tracing, scheduling, and prioritization.

Practice Description

Security requirements are often identified during the system life cycle. However, the requirements tend to be general mechanisms such as password protection, firewalls, virus detection tools, and the like. Often the security requirements are developed independently of the rest of the requirements engineering activity, and hence are not integrated into the mainstream of the requirements activities. As a result, security requirements that are specific to the system and that provide for protection of essential services and assets are often neglected.

In reviewing requirements documents, we typically find that security requirements, when they exist, are in a section by themselves and have been copied from a generic set of security requirements. The requirements elicitation and analysis that is needed to get a better set of security requirements seldom

21. #refs
22. #refs
23. #refs

takes place.

Much requirements engineering research and practice has addressed the capabilities that the system will provide. So a lot of attention is given to the functionality of the system, from the user's perspective, but little attention is given to what the system should *not* do [Bishop 02[28]]. In one discussion on requirements prioritization for a specific large system, ease of use was assigned a higher priority than security requirements. Security requirements were in the lower half of the prioritized requirements. This occurred in part because the only security requirements that were considered had to do with access control.

Current research recognizes that security requirements are negative requirements. As such, general security requirements, such as "the system shall not allow successful attacks" are generally not feasible, as there is no consensus agreement on ways to validate them other than to apply formal methods to the entire system, including COTS components. We can, however, identify the essential services and assets that must be protected. We are able to validate that mechanisms such as access control, levels of security, backups, replication, and policy are implemented and enforced. We can also validate that the system will properly handle specific threats identified by a threat model and correctly respond to intrusion scenarios.

If security requirements are not effectively defined, the resulting system cannot be effectively evaluated for success or failure prior to implementation. Security requirements are often missing in the requirements elicitation process, and tend to be neglected subsequently.

In addition to employing applicable software engineering techniques, the organization must understand how to incorporate the techniques into its existing software development processes [Linger 98[29]]. The identification of organizational mechanisms that promote or inhibit the adoption of security requirements elicitation can be an indicator of the security level of the resulting product.

An earlier report, focusing on survivable requirements engineering, provided background material [Mead 03a[30]]. By assembling an elicitation framework based on initial research and applying it to a software development effort, the Survivable Systems Engineering team at the Software Engineering Institute's CERT Coordination Center was able to identify additional research areas and refine the framework through further research.

If usable approaches to security requirements engineering continue to be developed and mechanisms to promote organizational use are identified, an organization can ensure that the resulting product effectively meets security requirements. In this content area, we discuss the Comprehensive, Lightweight Application Security Process (CLASP[31]) approach to security requirements engineering, System Quality Requirements Engineering (SQUARE[32]), and core security requirements artifacts[33]. We also discuss the use of attack trees[34] in security requirements engineering, and misuse/abuse cases[35] [process diagram[36]]. Formal specification approaches to security requirements, such as REVEAL[37] and

28. #refs
29. #refs
30. #refs
31. daisy:229 (Comprehensive Lightweight Application Security Process (CLASP))
32. daisy:232 (SQUARE Process)
33. http://computing-reports.open.ac.uk/index.php/2004/200423
34. daisy:236 (Attack Trees)
35. daisy:125 (Misuse and Abuse Cases: Getting Past the Positive)
36. daisy:245
37. http://www.praxis-his.com/reveal/index.htm

Software Cost Reduction (SCR[38]), have also been useful. The higher levels of the Common Criteria[39] provide similar results. In each section we list local references. A more comprehensive bibliography[40] is also included for this topic.

Although much work remains to be done, organizations can significantly improve the security of their systems by utilizing a systematic approach to security requirements engineering. The methods described here can help in this task.

Business Case Rationale

Although data exists to support the benefit of requirements engineering in general, the data to specifically support the benefits of security requirements engineering is anecdotal. Organizations that systematically develop security requirements see benefit from this activity, but it is not yet quantified in terms of return on investment. We hope that in the future more supporting data will be amassed and made available to support this important activity. Discussion of a broader business case development model should be helpful to those striving to develop specific business cases. [Business Relevance[43]]

Maturity of Practices

The techniques described have all had successful pilots and prototypes. SCR, REVEAL, and Common Criteria are mature practices.

References

[Bishop 02]	Bishop, Matt. *Computer Security: Art and Science*. Boston, MA: Addison-Wesley Professional, 2002.
[Boehm 88]	Boehm, Barry W. & Papaccio, Philip N. “Understanding and Controlling Software Costs. *IEEE Transactions on Software Engineering 14*, 10 (October 1988): 1462-1477.
[Charette 05]	Charette, R. N. “Why Software Fails.” *IEEE Spectrum 42*, 9 (September 2005): 42-29.
[Jones 86]	Jones, Capers, ed. *Tutorial: Programming Productivity: Issues for the Eighties, 2nd Ed*. Los Angeles: IEEE Computer Society Press, 1986.
[Linger 98]	Linger, R. C.; Mead, N. R.; & Lipson, H. F. "Requirements Definition for Survivable Systems," 14-23. *Third International Conference on Requirements Engineering*. Colorado Springs, CO, April 6-10, 1998. Los Alamitos, CA: IEEE Computer Society, 1998.
[McConnell 01]	McConnell, Steve. “From the Editor - An Ounce

38. http://www.softwaretechnews.com/stn3-4/scr.html
39. daisy:239 (The Common Criteria)
40. daisy:231 (Requirements Engineering Annotated Bibliography)
43. daisy:74 (Business Case)

of Prevention.” *IEEE Software 18*, 3 (May 2001): 5-7.

[Mead 03a] Mead, N. R. *Requirements Engineering for Survivable Systems* (CMU/SEI-2003-TN-013). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003. http://www.sei.cmu.edu/publications/documents/03.reports/03tn0

[NIST 02] National Institute of Standards and Technology. "Software Errors Cost U.S. Economy $59.5 Billion Annually" (NIST 2002-10). http://www.nist.gov/public_affairs/releases/n02-10.htm (2002).

[Soo Hoo 01] Soo Hoo, Kevin; Sudbury, Andrew W.; & Jaquith, Andrew R. "Tangible ROI through Secure Software Engineering." *Secure Business Quarterly* 1, 2 (Fourth Quarter 2001). http://www.sbq.com/sbq/rosi/sbq_rosi_software_engineering.pdf

[Wiegers 03] Wiegers, Karl E. *Software Requirements*. Redmond, WA: Microsoft Press, 2003.

SEI Copyright

Fields

Name	Value
Copyright Holder	SEI

Fields

1. http://www.sei.cmu.edu/about/legal-permissions.html

Name	Value
is-content-area-overview	true
Content Areas	Best Practices/Requirements Engineering
SDLC Relevance	Requirements
Workflow State	Publishable